| DOI II DI                            | o i owio i i                     |                               |   |  | ptr-                     |   |  |          |
|--------------------------------------|----------------------------------|-------------------------------|---|--|--------------------------|---|--|----------|
| DSA-II PRACTICAL                     |                                  | }                             |   | >rchild=temp;                              | <del>-</del>             |   | <pre>printf("\nPreorder traversal: ");</pre> |          |
| SHORT NOT                            | ΓES (ANDY)                       |                               |   |  | break;                   |   |  |          |
| . Dina                               | w. Coareh                        | else                          |   |  | }                        | preorder(ro                                   | ot);   |          |
|                                      | ry Search                        |                               |   |  | else                     |   | break;                                       |          |
| Tree                                 | ( BST ):                         | ptr=ptr                       | ->rchild;   | S  | ptr=ptr-                 | case 4:                                       |  |          |
| #include <stdio.h></stdio.h>         |                                  | }<br>}//while                 |   | >rchild;                                   |                          | printf("\nPostorder Traversal: ");            |  |          |
| #include <stdlib.h></stdlib.h>       |                                  | }//else                       |   | ,  |                          | pring (mi osioraei iraveisai.),               |  |          |
| struct node                          |                                  |                               | \n Add More   | return(root);                              |                          | postorder(root);                              |  |          |
| {                                    |                                  | (Y/N)?:");                    |   | }  | }                        |   | break;                                       |          |
| struct node *lchild;                 |                                  | scanf(" %c",&ch);             |   |  |                          |   | case 5: printf("\nEnter the Value            |          |
| int data;                            |                                  | }while(ch=='Y'    ch=='y');   |   | return(root);                              | return(root);            |   | ed: ");                                      |          |
| struct node *rchild;                 |                                  | return(root);                 |   | }  |                          |   |  |          |
| };                                   |                                  | }                             |   |  |                          | scanf("%d'                                    | ',#);  |          |
| typedef struct node NODE;            |                                  | int search(int num,NODE *ptr) |   | · · · · · · · · · · · · · · · · · · ·      |                          |   |  |          |
| NODE *getnode()                      |                                  | {                             |   | void inorder(NODE *ptr)                    |                          | t=search(ni                                   |  |          |
| {                                    |                                  | while(ptr!=NULL)              |   | {  |                          |   | if(t==1)                                     |          |
| NODE *temp;                          |                                  | \<br>if(num==ptr->data)       |   | if(ptr!=NULL)                              |                          | nrintf("\nV                                   | alue is found");                             |          |
| •                                    |                                  | return 1;                     |   | '<br>inorder(ptr->lchild);                 |                          | pring( mr                                     | else   |          |
| temp=(NODE*)mo                       | alloc(sizeof(NODE))              | retur                         | и 1,  | printf(" %d",ptr->                         |                          |   | eise   |          |
| ;                                    |                                  | if(num <ptr->data)</ptr->     |   | inorder(ptr->rchild);                      |                          |   | printf("\nValue is not                       |          |
| printf("\n\n Ent                     | printf("\n\n Enter the data: "); |                               | 1   |  | 1                        |   | found");                                     |          |
| scanf("%d",&te                       | emp->data);                      |                               | ·   | í  |                          | <b>,</b> , ,                                  | break;                                       |          |
| temp->lchild=N                       | ULL;                             | ptr=ptr-                      | >lchild;  | ,  |                          | case 6:                                       | ,  |          |
| temp->rchild=N                       | ULL;                             |                               |   |  |                          |   |  |          |
| return(temp);                        |                                  |                               | }   | void preorder(NODE                         | *ptr)                    |   | temp=getnode();                              |          |
| }                                    |                                  |                               |   | {  |                          |   |  |          |
| NODE *create()                       |                                  | if(num>ptr->data)             |   | if(ptr!=NULL)                              |                          |   | root=insert(temp,root);                      |          |
| {                                    |                                  |                               | {   | {  |                          |   | break;                                       |          |
| NODE *temp,*p                        | otr,*root;                       |                               |   | printf(" %d",ptr->c                        | data);                   | case 7:                                       |  |          |
| char ch;                             |                                  | ptr=ptr->rchild;              |   | preorder(ptr->lchild);                     |                          |   | exit(1);                                     |          |
| root=NULL;                           |                                  |                               | }   | preorder(ptr->rchit                        | ld);                     | }   |  |          |
| do                                   |                                  |                               |   | }  |                          | }   |  |          |
| i tam                                | p=getnode();                     |                               |   | }  |                          | }   |  |          |
|                                      |                                  | }                             |   | void postorder(NODE *ptr)                  |                          | • A   | djacency Matrix                              |          |
| if(root==NULL)<br>root=temp;<br>else |                                  | return 0;                     |   | { if(ptr!=NULL)                            |                          | with In degree ,<br>Out degree ,              |  |          |
|                                      |                                  |                               |   |  |                          |   |  | <i>f</i> |
| ptr=                                 | =root;                           | ,                             |   | postorder(ptr->lchi<br>postorder(ptr->rchi |                          | -   | otal acgree (                                |          |
| while(ptr!=NULL)                     |                                  | )<br>NODE *incart(NODE        | NODE *insert(NODE *temp, NODE   |  | printf(" %d",ptr->data); |   | #include <stdio.h></stdio.h>                 |          |
| {                                    | •                                |                               | temp, NODE  |  | uuiu),                   | #include <st< td=""><td>dlib.h&gt;</td></st<> | dlib.h>                                      |          |
|                                      | if(temp-                         | *root)<br>{                   |   | <i>;</i><br>}                              |                          | struct NOD                                    | E  |          |
| >data <ptr->data)</ptr->             |                                  | NODE *ptr;                    |   | ,  |                          | {   |  |          |
|                                      | {                                | ptr=root;                     |   | main()                                     |                          | int data;                                     |  |          |
|                                      |                                  | if(ptr==NULL)                 |   | {  |                          | struct NO                                     | DE *next;                                    |          |
| if(pti                               | r->lchild==NULL)                 | {                             |   | int ch,num,t,abc;                          |                          | };  | -4 NODE J                                    |          |
|                                      |                                  | root=ptr=temp;                |   | NODE *root;                                |                          | ** *  | ct NODE node;                                |          |
| {                                    |                                  | }                             |   | NODE *temp;                                |                          | node *list[1<br>int nov;                      | 0];  |          |
|                                      |                                  | else                          |   | while(1)                                   |                          |   | odenum(int vno)                              |          |
| S1 1711 /                            | ptr-                             | <i>(</i>                      |   | {  |                          | f   | acriam (iii 7110)                            |          |
| >lchild=temp;                        |                                  | ptr=root;                     |   | printf("\nMain Me                          |                          | node *tem                                     | p;   |          |
|                                      | break;                           | while(ptr!=NULL)              | )   | printf("\n1: Create                        | Binary search            |   | de*) malloc(sizeof(node));                   |          |
|                                      | Dreun,                           | {                             |   | tree");                                    |                          | temp->da                                      | a=vno;                                       |          |
| }                                    |                                  |                               | ->data <ptr-< td=""><td>printf("\n2: Inorde</td><td></td><td>temp-&gt;ne.</td><td>ct=NULL;</td></ptr-<> | printf("\n2: Inorde                        |                          | temp->ne.                                     | ct=NULL;                                     |          |
| ,                                    |                                  | >data)                        |   | printf("\n3: Preora                        |                          | return ten                                    | ıp;  |          |
| else                                 |                                  | {<br>:6/ntu                   |   | printf("\n4: postor                        |                          | }   |  |          |
|                                      |                                  | if(ptr-<br>>lchild==NULL)     | •   | printf("\n5: Search<br>printf("\n6: Insert |                          | void display                                  | (node *list[10])                             |          |
| ptr                                  | =ptr->lchild;                    | {                             |   | printf("\n7: Exit")                        |                          | 1   |  |          |
|                                      | }                                | ,                             | ptr-  | printf("\n Enter the                       |                          | int i;  |  |          |
|                                      | else                             | >lchild=temp;                 | •   | scanf("%d",&ch);                           |                          | node *ptr;                                    |  |          |
|                                      | {                                | r/                            | break;  | switch(ch)                                 |                          | for(i=1;i<=                                   | =nov;i++)                                    |          |
|                                      |                                  | }                             |   | {  |                          | <i>{</i>                                      |  |          |
| if(pt                                | tr->rchild==NULL)                | else                          |   | case 1: root=crea                          | ate();                   |   | printf("V%d ",i);                            |          |
|                                      |                                  |                               | ptr=ptr-  |  | break;                   | Conference 12 11                              | il-ntel=NIII I                               |          |
| {                                    |                                  | >lchild;                      |   | case 2:                                    |                          | for(ptr=list[<br>>next)                       | i];ptr!=NULL;ptr=ptr-                        |          |
|                                      |                                  | }                             |   |  |                          | ~next)  |  |          |
| ptr-                                 |                                  | else                          |   | printf("\nInorder traversal");             |                          | nrintf("%d.                                   | >",ptr->data);                               |          |
| >rchild=temp;                        |                                  | {                             |   |  |                          | F 9 ( / / / / /                               | printf("NULL");                              |          |
|                                      | break;                           |                               | if(ptr-   | inorder(root);                             |                          |   | printf("\n");                                |          |
|                                      | vieuk,                           | >rchild==NULL)                |   |  | break;                   | <b>?</b>                                      | - * * **                                     |          |

case 3:

```
if(ptr-
void creatadjacencylist()
                                                    printf("\nEnter no. of vertices: ");
                                                                                                                                                         >lchild==NULL && ptr-
                                                    scanf("%d",&nov);
                                                                                                                                                         >rchild==NULL)
                                                                                                                                break;
             int i,j;
             char ch;
                                                    creatadjacencylist();
                                                                                                                                                                      leaf total ++;
                                                    printf("\n*****Adjacency
             node *temp, *last;
             for(i=1;i<=nov;i++)
                                                   List******|n");
                                                                                                                                                                      leafcount(ptr->rchild);
                                                    display(list);
             list[i]=NULL;
                                                   degree();
                                                                                                                                                                      return leaftotal;
                                                                                                                   else
             for(i=1;i<=nov;i++)
                                                             Leaf Node:
                                                                                                                                                         main()
                                                                                                                                ptr=ptr-
             for(j=1;j<=nov;j++)
                                                                                                      >lchild;
                                                   #include<stdio.h>
                                                                                                                                                                      NODE *root:
                                                   #include<stdlib.h>
                                                                                                                                                                      root=create();
             printf("\nIs there edge
                                                   int nodetotal=0,leaftotal=0;
                                                                                                                                                                      printf("\n");
between V[%d] and V[%d] (choose:
                                                                                                                                                                      nodetotal=totalnode(root
                                                                                                                                else
y/n): ",i,j);
                                                                                                                                                         );
                                                   struct node
                                                                                                                                                                      leaf total = leaf count (root);\\
                                                   {
             scanf(" %c",&ch);
                                                                                                                                                                      printf("\n The total no.
                                                                struct node *lchild;
                                                                                                                                                         of nodes are: %d",nodetotal);
                                                                int data:
             if(ch=='Y' \mid\mid ch=='y')
                                                                                                                   if(ptr->rchild==NULL)
                                                                                                                                                                      printf("\n\n The total no.
                                                                struct node *rchild;
                                                                                                                                                         of leaf nodes are: %d",leaftotal);
                                                   typedef struct node NODE;
                                                                                                                                                                   Adjacency List:
                                                   NODE *getnode()
             temp=getnodenum(j);
                                                                NODE *temp;
                                                                                                                                ptr-
                                                                                                                                                         #include<stdio.h>
                                                                                                      >rchild=temp;
                                                                                                                                                         #include<stdlib.h>
                                                                temp=(NODE*)malloc(si
             if(list[i]==NULL)
                                                   zeof(NODE));
                                                                                                                                                         struct NODE
                                                                printf("\n\n Enter the
                                                   data : ");
                                                                                                                                                          int data;
             list[i]=temp;
                                                                scanf("%d",&temp-
                                                                                                                                                          struct NODE *next;
                                                   >data);
                          else
                                                                                                                                                         typedef struct NODE node;
                                                                temp->lchild=NULL;
                                                                temp->rchild=NULL;
                                                                                                                                                         node *list[10];
                                                                return(temp);
                                                                                                                                                         int nov;
                                                                                                                   else
                                                   }
                                                                                                                                                         node *getnodenum(int vno)
             for(last=list[i];last-
                                                   NODE *create()
                                                                                                                                                           node *temp;
>next!=NULL;last=last->next);
                                                                                                                                ptr=ptr-
                                                                                                                                                          temp=(node*) malloc(sizeof(node));
                                                                                                      >rchild:
                                                                NODE *temp, *ptr, *root;
                                                                                                                                                           temp->data=vno;
                                                                char ch:
                                                                                                                                                          temp->next=NULL:
             last->next=temp;
                                                                root=NULL;
                                                                                                                                                          return temp;
                                                                do
                                                                                                                   }//while
                                                                                                                                                         void display(node *list[10])
                                                                                                                                } //else
                                                                                                                                printf("\n
                                                                temp=getnode();
                                                                                                                                                          int i;
                                                                                                      Add More (Y/N)? : ");
                                                                                                                                                          node *ptr;
                                                                if(root==NULL)
                                                                                                                                scanf("
                                                                                                                                                          for(i=0;i<nov;i++)
                                                                                                      %c",&ch);
                                                                                                                   }while(ch=='Y' ||
                                                                                                                                                                       printf("V%d ",i+1);
                                                                root=temp;
                                                                                                      ch=='y');
                                                                             else
void degree()
                                                                                                                   return(root);
                                                                                                                                                         for(ptr=list[i];ptr!=NULL;ptr=ptr-
                                                                                                                                                         >next)
  int i,cnt,outcnt[10],incnt[10]={0};
  node *ptr;
                                                                                                      int totalnode(NODE *ptr)
                                                                                                                                                         printf("%d->",ptr->data);
  for(i=1;i<=nov;i++)
                                                                while(ptr!=NULL)
                                                                                                                                                                       printf("NULL");
                                                                                                                   if(ptr!=NULL)
   1
                                                                                                                                                                       printf("\n");
for(ptr=list[i],cnt=0;ptr!=NULL;ptr=pt
r->next,cnt++)
                                                                                                                   nodetotal++;
                                                                             if(temp-
      incnt[ptr->data]+=1;
                                                                                                                                                         void creatadjacencylist()
                                                   >data<ptr->data)
    outcnt[i]=cnt;
                                                                                                                   totalnode(ptr->lchild);
                                                                                                                                                                      int i,j;
  printf("\nVertex\t Indegree\t
                                                                                                                   totalnode(ptr->rchild);
                                                                                                                                                                      char ch;
Outdegree\t Totaldegree");
                                                                                                                                                                      node *temp, *last;
  for(i=1;i<=nov;i++)
                                                                                                                   return nodetotal;
                                                                if(ptr->lchild==NULL)
                                                                                                                                                                      for(i=0;i<nov;i++)
   printf("\nV%d \t\t %d \t\t %d \t\t
%d",i,incnt[i],outcnt[i],incnt[i]+outcnt
                                                                                                      int leafcount(NODE *ptr)
                                                                                                                                                                      list[i]=NULL;
[i]);
                                                                                                                                                                      for(i=0;i<nov;i++)
                                                                                                                   if(ptr!=NULL)
?
main()
                                                                                                                                                                      for(j=0;j<nov;j++)
                                                                             ptr-
 int i,j,a[20][20];
                                                                                                                   leafcount(ptr->lchild);
                                                   >lchild=temp;
                                                                                                                                                                                   {
```

```
NODE *getnode(int j)
                                                                                                                                                                         printf("\nAdjacency
             printf("\nIs there edge
                                                     int i;
                                                                                                                                                            matrix:- \n");
                                                                                                         NODE * temp;
between V[%d] and V[%d] (Choose
                                                     for(i=0;i<MAX;i++)
                                                                                                                                                                         display(ver,matric);
(y/n): ",i+1,j+1);
                                                      q \rightarrow Q[i] = 0;
                                                                                                         temp=(NODE
                                                                                                                                                                         getch();
                                                     q->rear=-1;
                                                                                                        *)malloc(sizeof(NODE));
                                                                                                                                                                         adjmat(ver,matric);
             scanf(" %c",&ch);
                                                     q->front=-1;
                                                                                                         temp->data=j;
                                                                                                                                                                         getch();
                                                     printf("\nQueue created");
                                                                                                         temp->link=NULL;
                                                                                                                                                                         bfs(ver,matric);
             if(ch=='Y' \mid\mid ch=='y')
                                                                                                         return(temp);
                                                                                                                                                                         getch();
                                                   void add(int data)
                                                                                                        NODE *findlast(NODE *h)
                                                                                                                                                                      DFS:
                                                     q->Q[++q->rear]=data;
             temp=getnodenum(j+1);
                                                                                                         NODE *ptr;
                                                                                                                                                            #include<stdio.h>
                                                                                                         for(ptr=h;ptr->link!=NULL;ptr=ptr-
                                                                                                                                                            int nov,a[20][20];
                                                                                                        >link);
                                                                                                                                                            int visited[20];
             if(list[i]==NULL)
                                                   int delet()
                                                                                                         return(ptr);
                                                                                                                                                            void creatematrix()
                                                                                                                                                                         int i,j;
             list[i]=temp;
                                                     return(q->Q[++q->front]);
                                                                                                        void displaylist(NODE *h[10],int n)
                                                                                                                                                                         printf("\nEnter no. of
                                                                                                                                                            vertices: ");
                          else
                                                   int isempty()
                                                                                                         NODE *ptr;
                                                                                                                                                                         scanf("%d",&nov);
                                                                                                         int i;
                                                                                                         for(i=0;i<n;i++)
                          1
                                                     if (q->rear==q->front)
                                                                                                                                                                         //accept the matrix
                                                      return 1;
                                                                                                                                                                         for(i=1;i<=nov;i++)
                                                                                                           printf("\n V%d ",i+1);
                                                     else
             for(last=list[i];last-
                                                     return 0;
                                                                                                           ptr=h[i];
>next!=NULL;last=last->next);
                                                                                                           if(ptr==NULL)
                                                                                                                                                                         for(j=1;j<=nov;j++)
                                                                                                            printf(" NULL");
                                                                                                                                                                                      {
                                                   void accept(int n,int a[MAX][MAX])
                                                                                                           while(ptr!=NULL)
             last->next=temp;
                                                                                                                                                                         printf("\nIs there egde
                                                                                                                     printf(" -> %d",ptr-
                                                                 int i,j;
                                                                                                                                                            between V[%d] and V[%d]: ",i,j);
                                                                 for(i=0;i < n;i++)
                          ?
                                                                                                        >data);
                                                                                                                     ptr=ptr->link;
                                                                                                                                                                         scanf("%d",&a[i][j]);
                                                       for(j=0;j<n;j++)
                                                                                                            3
                                                                                                                                                                                      }
                                                                                                           printf("\n");
                                                       {
                                                                 printf("\n Enter
                                                                                                          }
                                                   a[%d][%d]: ",i+1,j+1);
             //display(list);
                                                                 scanf("%d",&a[i][j]);
                                                                                                        void adjmat(int n,int a[MAX][MAX])
                                                                                                                                                            void display(int a[20][20]) //print the
                                                        3
                                                                                                                                                            matrix
                                                      }
                                                                                                         NODE *ptr, *temp;
main()
                                                                                                         int i,j;
                                                                                                                                                                         int i,j;
                                                                                                         for(i=0;i<n;i++)
                                                   void display(int n,int a[MAX][MAX])
                                                                                                                                                                         for(i=1;i<=nov;i++)
 int i,j,a[20][20];
                                                                                                          h[i]=NULL;
                                                                 int i,j;
                                                                                                         for(i=0;i<n;i++)
printf("\nEnter no. of vertices: ");
                                                                 for(i=0;i<n;i++)
                                                                                                          1
                                                                                                                                                                         for(j=1;j<=nov;j++)
 scanf("%d",&nov);
                                                                                                           for(j=0;j<n;j++)
                                                       for(j=0;j<n;j++)
                                                                                                                                                                         printf("\t%d",a[i][j]);
creatadjacencylist();
                                                                                                                      if(a[i][j]!=0)
printf("\n*****Adjacency
                                                                 printf("\t %d ",a[i][j]);
                                                                                                                                                                         printf("\n");
List******|n");
                                                                                                                       temp = getnode(j+1);
 display(list);
                                                       printf("\n");
                                                                                                                       if(h[i]==NULL)
                                                                                                                        h[i]=temp;
                                                                                                                                                            void recdfs(int a[20][20],int nov,int
                                                                                                                       else
                                                   void bfs(int n,int a[MAX][MAX])
         BFS:
                                                                                                                         ptr=findlast(h[i]);
                                                                                                                                                                         int i;
                                                     int i,j,v[10]={0};
                                                                                                                         ptr->link=temp:
#include<stdio.h>
                                                                                                                                                                         visited[ver]=1;
                                                     printf("\n \t BFS seq. :\n ");
#define MAX 20
                                                                                                                                                                         printf(" V%d",ver);
                                                     i=1:
                                                                                                                                                                         for(i=1;i<=nov;i++)
                                                     add(i);
                                                                                                           3
struct n
                                                      v[i-1]=1;
                                                                                                         printf("\n The Adjacency list looks
                                                      while(isempty()!=1)
 int data;
                                                                                                        like ... \n");
 struct n *link:
                                                                                                                                                                         if((a[ver][i] == 1) \ \& \&
                                                                 i=delet();
                                                                                                         display list(h,n);
                                                                                                                                                            (visited[i]==0))
};
                                                                 for(j{=}\theta;j{<}n;j{+}{+})
typedef struct n NODE;
NODE *h[20];
                                                                   if(a[i-1][j]==1 &&
                                                                                                                                                                         recdfs(a,nov,i);
                                                    v[j]==0
struct queue
                                                                                                        void main()
                                                                               add(j+1);
int front,rear;
                                                                               v[j]=1;
                                                                                                         int ver,matric[MAX][MAX];
int Q[MAX];
                                                                                                                                                            main()
                                                                                                                     clrscr();
                                                                                                                     printf("\n\t\t\t Enter the
typedef struct queue QUEUE;
                                                                                                                                                                         int ch,i;
                                                                 printf("\t V%d ",i);
                                                                                                        no. of vertex ");
QUEUE *q;
                                                                                                                                                                         creatematrix();
                                                                                                                     scanf("%d",&ver);
                                                                                                                                                                         printf("\n\t***Adjacency
                                                                                                                     accept(ver,matric);
void initqueue()
                                                                                                                                                            Matrix **** \n");
```

```
display(a);
                                               Node* newNode(int item)
                                                                                                           root = insert(root, 4);
                                                                                                           root = insert(root, 7);
    printf("\nThe Depth First search
                                                           Node* temp = new Node;
                                                                                                           root = insert(root, 6);
Traversal(DFS) is:");
                                                           temp->key = item;
                                                                                                           root = insert(root, 8);
    recdfs(a,nov,1);
                                                            temp->left = temp->right
                                               = NULL:
                                                                                                           oddNode(root);
         Heap Sort (by
                                                            return temp;
                                                                                                           return 0;
         putting value in
         int main() ):
                                               // A utility function to do inorder
                                               traversal of BST
#include <stdio.h>
                                               void inorder(Node* root)
void swap(int* a, int* b)
                                                                                               Even
                                                           if (root != NULL) {
int temp = *a;
*a = *b;
                                                                                               #include <stdio.h>
                                                            inorder(root->left);
*b = temp;
                                                                       printf("%d
                                                                                               #include <stdlib.h>
                                                ", root->key);
                                                                                               struct node
void heapify(int arr[], int N, int i)
                                                           inorder(root->right);
                                                                                                  int key;
int largest = i;
int \ left = 2 * i + 1;
                                                                                                  struct node *left,
int\ right = 2*i + 2;
                                                                                               *right;
if (left < N \&\& arr[left] > arr[largest])
                                                                                               };
                                               /* A utility function to insert a new
largest = left;
if (right < N & & arr[right] >
                                               with given key in BST */
                                                                                               struct node
arr[largest])
                                               Node* insert(Node* node, int key)
                                                                                               *newNode(int item) {
largest = right;
                                                                                               struct node *temp=(struct
if (largest != i)
                                                           /* If the tree is empty,
                                                                                               node
                                               return a new node */
swap(&arr[i], &arr[largest]);
                                                                                               *)malloc(sizeof(struct
                                                           if (node == NULL)
heapify(arr, N, largest);
                                                                                               node));
                                                                                                   temp->key = item;
                                               newNode(key);
                                                                                                   temp->left = temp-
void heapSort(int arr[], int N)
                                                            /* Otherwise, recur down
                                                                                               >right = NULL;
                                               the tree */
                                                                                                  return temp;
for (int i = N/2 - 1; i >= 0; i--)
                                                            if (key < node->key)
heapify(arr, N, i);
                                                                        node->left
for (int i = N - 1; i >= 0; i--) {
                                               = insert(node->left, key);
swap(&arr[0], &arr[i]);
                                                                                               struct node* insert(struct
                                                           else
heapify(arr, i, 0);
                                                                                               node* node, int key) {
                                                                        node->right
                                                                                                  if (node == NULL)
                                               = insert(node->right, key);
                                                                                               return newNode(key);
void printArray(int arr[], int N)
                                                           /* return the
                                                                                                   if (key < node->key)
                                               (unchanged) node pointer */
                                                                                                      node->left =
for (int i = 0; i < N; i++)
                                                            return node;
                                                                                               insert(node->left, key);
printf("%d ", arr[i]);
printf("\n");
                                                                                                  else
                                                                                                      node->right =
                                               // Function to print all odd nodes
int main()
                                                                                               insert(node->right, key);
                                               void oddNode(Node* root)
                                                                                                   return node;
                                                            if (root != NULL) {
{10,50,60,40,30,20,80,90,70,100};
int N = sizeof(arr) / sizeof(arr[0]);
                                                            oddNode(root->left);
                                                                                               void main(void)
heapSort(arr, N);
printf("Sorted array is\n");
                                                                       // if node is
                                                                                                  struct node *root =
printArray(arr, N);
                                               odd then print it
                                                                                               NULL:
                                                                        if (root-
                                                                                                  int
                                               >key % 2 != 0)
         BST sum odd() &
                                                                                               data[]={3,1,2,6,4,7,8}, n,
         sum even ():
                                                           printf("%d ", root->key);
// C++ program to print all odd node of
                                                                                               n=sizeof(data)/sizeof(dat
BST
                                                           oddNode(root->right);
#include <bits/stdc++.h>
                                                                                                  for(i=0;i< n;i++)
using namespace std;
                                                                                               root=insert(root,data[i]);
// create Tree
                                               // Driver Code
struct Node {
                                               int main()
            int kev:
            struct Node *left, *right;
                                                            Node* root = NULL:
                                                                                                       *****
                                                            root = insert(root, 5);
// A utility function to create a new
                                                            root = insert(root, 3);
BST node
                                                            root = insert(root, 2);
```